RIASSUNTO SLIDE PER SCRITTO SISTEMI INFORMATIVI – Capitoli 3,4,5,6

Capitolo 3 – Pianificazione dei SI

Ruolo dell'analista di sistemi

L'**analista di sistemi** svolge un ruolo chiave nello sviluppo dei sistemi informativi, analizza la situazione di business di un'azienda, identifica le opportunità di miglioramenti e progetta SI per implementare i miglioramenti individuati.
L'analista interagisce con numerosi professionisti: specialisti tecnici (DBA, network admin, programmatori), persone dell'area business (utenti, dirigenti etc.).
Esistono vari ruoli specialistici per un'analista:

- people oriented -> si occupa del project management
- Business oriented -> analista dei requisiti e business Analyst
- Technically-oriented -> analista delle infrastrutture
- Generalista -> analista di sistemi

Un analista di sistemi deve avere delle skill di problem solving buone, carattere ed etica, conoscenza delle tecnologie informatiche più recenti, esperienza nella computer programming, conoscenze generale di business.

System development life cycle (SDLC)

Il ciclo di vita di un Sistema è compost dalle seguenti fasi:

- Pianificazione (capitolo 3): avvio del progetto, si prepara una *System request* e si esegue un'analisi preliminare di fattibilità. In questa fase rientra anche la predisposizione del progetto: si costruisce il *project plan* includendo una pianificazione dei lavori e delle risorse necessarie per realizzare il progetto.
- Analisi (capitolo 4): si parte individuando una strategia di analisi studiando il sistema attuale ed i suoi problemi, vengono raccolti e analizzati i requisiti per il nuovo sistema descrivendolo con modelli di analisi. Infine, viene preparata e presentata la nuova proposta di sistema dove sono riassunti i risultati della fase di analisi.
- Progettazione (capitolo 5): si sceglie la strategia di acquisizione del sistema informativo (se costruirlo o comprarlo), si progettano i componenti del sistema (architettura, interfaccia, database, programmi)

Il dilemma “make, buy or customize”

	Buy	Customize	Make
Requisiti	Approssimati	Pochi	Esatti
Modificabilità	Difficile	Elevata	Buona
Costo	Dipende da requisiti e diffusione	Dipende	Alto

- Implementazione (capitolo 6): il sistema viene costruito, testato ed installato. In questa fase è incluso anche il supporto al sistema in funzione

Avvio del progetto: BPM, PROJECT SPONSOR, SYSTEM REQUEST E ANALISI FATTIBILITÀ

I progetti sui SI provengono da un bisogno del Business che può essere l’abilitazione ad una iniziativa o nuova strategia di business, il supporto di una fusione/acquisizione, la risoluzione di un punto critico o l’implementazione di una nuova tecnologia oppure una conseguenza del **BPM** (**business process management**).
Il **BPM** è un ciclo continuo a quattro fasi:

1. Definire e mappare i passi di un processo aziendale.
2. Creare modi per migliorare i passi nel processo che aggiungono valore.
3. Individuare i modi per eliminare o consolidare i passi nel processo che non aggiungono valore.
4. Creare o adattare i workflow informatici per farli corrispondere alle mappe migliorate del processo.

Il BPM identifica i bisogni di un business, si occupa:

- Automazione dei processi di business: creazione o adattamento dei workflow informatici per farli corrispondere alle mappe migliorate del processo
- Miglioramento dei processi di business: studio dei processi di business, creazione o riprogettazione di nuovi per migliorare i workflow dei processi e/o utilizzo di nuovo tecnologie che abilitano nuove strutture dei processi
- Reengineering dei processi di business: una totale revisione dei processi

Una forte esigenza di business conduce una persona o un gruppo a farsi avanti come **project sponsor**: è/sono il motore dietro il progetto, specifica i requisiti di business complessivi e determina il business value. Per l’inizializzazione il project sponsor selezione il giusto project manager per il lavoro, si assicura dell’organizzazione appropriata del lavoro e garantisce tempo sufficiente per svolgere le attività iniziali propriamente. Partecipa ai kick-off meeting e si occupa delle decisioni go/no go per lo stage successivo. Per la pianificazione controlla che i piani siano realistici e approva solo quelli realizzabili. Si assicura che il team non è forzato ad impegnarsi

per aspettative irrealistiche e garantisce tempo a sufficienza per la pianificazione del progetto. Durante la fase di implementazione lavora con il project manager e non oltrepassa i confini concentrandosi troppo sui dettagli, valuta i progressi rispetto agli obiettivi e guida il project manager esprimendo il proprio feedback. Motiva i project manager e i membri del team ad indentificare e risolvere i loro problemi con il progetto. Nella fase di chiusura partecipa alle valutazioni post-project, valuta la performance del progetto in base all'aderenza con le specifiche ed i criteri di successo.
Il project sponsor richiede formalmente un progetto tramite una **system request** che descrive le ragioni business per il progetto e definisce il valore atteso per il sistema ed elenca gli elementi chiave del progetto, forzando lo sponsor a formalizzare le sue idee e fornire un framework per raccogliere le informazioni iniziali sul progetto e inoltre standardizza l'informazione utilizzata dalla commissione d'approvazione del progetto.
Gli elementi della system request sono:

- *Project sponsor*: la persona che inizia il progetto e svolge il ruolo di contatto primario per il progetto sul lato business
- *Business Need;* le ragioni legate al business per iniziare il sistema
- *Business Requirements:* le nuove (o migliorate) capacità lato business che il sistema offrirà
- *Business Value:* I benefici che il sistema creerà per l'organizzazione, per stimare il valore occorre identificare le fonti di guadagno (aumento delle vendite, riduzione dei costi, meno personale) ed assegnare dei valori come stime iniziali.
- *Special Issues or Constraints:* questioni che appartengono all'approvazione della commissione di decisione

L'**analisi di fattibilità** è un business case dettagliato per il progetto, compilato in uno studio di fattibilità, si divide in:

- *Fattibilità tecnica,* si analizzano le fonti di rischio tecniche come la mancanza di familiarità di utenti e analisti con l'area applicativa di business, mancanza di familiarità con le tecnologie, la dimensione del progetto e la compatibilità con sistemi esistenti (viene spesso richiesto un grado di integrazione).
- *Fattibilità economica* si identificano i costi ed i benefici, assegnando dei valori ad entrambi, si determina il cash flow e si valuta la sostenibilità finanziaria (ROI ovvero il ricavato dagli investimenti, BEP ovvero punto di pareggio, NPV

ovvero valore attuale netto)

$$\text{ROI} = \frac{\text{Total Benefits} - \text{Total Costs}}{\text{Total Costs}}$$

$$\text{BEP} = \begin{array}{c}\text{Number of}\\ \text{years of}\\ \text{negative}\\ \text{cash flow}\end{array} + \frac{\text{That year's Net Cash Flow} - \text{That year's Cumulative Cash Flow}}{\text{That year's Net Cash Flow}}$$

(*In the year in which Cumulative Cash Flow turns positive*):

$$\text{PV} = \frac{\text{Cash flow amount}}{(1 + |\text{rate of return})^n}$$ where n is the year in which the cash flow occurs.

- *Fattibilità organizzativa,* si osserva se gli obiettivi del progetto sono allineati con la business strategy -> un allineamento strategico stretto con la strategia di business migliora la probabilità di successo. Viene valutato l'effetto sui diversi stakeholders, i gruppi di stakeholder possono essere influenzati mediante presentazioni che descrivono e promuovono i benefici oppure attraverso dei prototipi che aiutano a dimostrare il concetto di sistema. Inoltre, un vero coinvolgimento degli utenti durante tutto il progetto può essere d'aiuto.

Dopo aver realizzato un'analisi della fattibilità esaustiva, occorre valutarla: tutti i progetti hanno dei rischi di fattibilità, l'obiettivo è conoscere i rischi da affrontare (analisi dei **SWOT:** strenghts, weaknesses, opportunities, threats) e la loro entità; una volta conosciuti i rischi si può cercare di mitigarli. L'analisi della fattibilità deve essere continuamente rivista durante il ciclo di vita del progetto.

Selezione del progetto: Project portfolio management, project plan, Sviluppo strutturato vs RAD vs ADM

Esistono diversi modi per caratterizzare i progetti: dimensione, costo, scopo, durata, ambito, rischio, valore economico etc.
L'approvazione di un progetto si basa sulla system request e sullo studio di fattibilità, per la selezione entra in gioco il project portfolio, ovvero bisogna valutare come si colloca il progetto entro l'intero portfolio di progetti? Inoltre, bisogna soddisfare una necessità di compromessi, ovvero selezionare progetti per formare un project portfolio bilanciato. Progetti sostenibili possono essere respinti o differiti a causa del project portfolio. Il **PPM (project portfolio management)** è un software che raccoglie e gestisce le informazioni su tutti i progetti, in corso ed in attesa di approvazione. Le aziende restano aggiornate sui progetti e si adattano al variare delle condizioni, alcune caratteristiche sono: prioritizzazione dei progetti, allocazione del personale, monitoraggio real-time dei progetti e segnalazione di variazione temporali o di costi.
Quando il progetto viene approvato, il responsabile del progetto deve: selezionare la migliore metodologia per il progetto, sviluppare un piano di lavoro per il progetto,

stabilire un piano per il personale e creare modi per coordinare e controllare il progetto.
Una **metodologia** è un approccio formalizzato per implementare il System Development Life Cycle, fonti utili per la metodologia possono essere sviluppate internamente da un'organizzazione oppure da una società di consulenza o da fornitori software. I fattori che possono influenzare la scelta di una determinata metodologia sono la familiarità con la tecnologia, complessità del sistema, l'affidabilità del sistema e la schedule visibility.
Esempi di metodologie:

- Uno **sviluppo strutturato** è basato sul **SDLC (System Development Life Cycle)**, prima di passare ad una fase successiva deve essere completata la precedente -> development a cascata, parallelo o V-model. Completare ogni fase prima di andare avanti assicura risultati corretti e di alta qualità (la rigidità imposta può essere uno svantaggio in condizioni dinamiche ed incerte).
- La **RAD** (**Rapid Application Development**) integra tecniche e strumenti speciali come CASE tools, JAD sessions, Visual programming languages o code generators. L'obiettivo consiste nello sviluppare parte del sistema per metterla nelle mani degli utenti. Esistono tre approcci per la RAD: l'*iterative development* in cui vengono sviluppate una serie di versioni sequenzialmente. Il *system prototyping* consiste nel creare un prototipo del sistema e crescerlo fino al sistema finale. Il throw-*away prototyping* è un metodo alternativo al design tramite prototipo in maniera sperimentale, la costruzione del sistema segue il design del prototipo ma abbandona il prototipo.

- La **ADM (agile development methodologies)** si focalizza su cicli brevi che producono un prodotto software completo, altamente adattabile in ambienti dinamici. Un esempio ne è l'extreme programming (XP), lo Scrum, ed altri.

Is There a "Central" Methodology?	*Most Common Project Management Methodologies (In Order)*	*Who Typically Leads Engineering Projects?*
Big Tech & Public Tech Companies		
- Teams can choose how they work (common) - Suggested methodology, but teams can choose (less frequent)	1. Plan->Build (iterate)->Ship 2. No "formal" methodology	- Tech lead - An engineer on the team
Venture-funded scaleups (Series B & above)		
- Suggested methodology, but teams can choose (common) - Teams expected to follow specific a methodology (less frequent)	1. Plan->Build (iterate)->Ship 2. No "formal" methodology 3. Kanban 4. Scrum	- Tech lead - An engineer on the team
Venture-funded startups (up to Series A)		
- Teams expected to follow specific a methodology (common) - Suggested methodology, but teams can choose (less frequent)	1. Plan->Build (iterate)->Ship 2. Scrum 3. Kanban	- An engineer on the team - Product manager
Non-venture funded tech companies		
- Teams expected to follow specific a methodology (mostly) - Suggested methodology, but teams can choose (less frequent)	1. Scrum 2. Others (Kanban, SAFe, Scaled Agile)	- Tech lead - Dedicated project manager - An engineer on the team
Large, non-tech companies		
- Teams expected to follow specific a methodology (mostly) - Suggested methodology, but teams can choose (rarely)	1. Scrum 2. SAFe 3. Others (Plan->build->ship, Kanban)	It varies: - A dedicated project manager - Scrum master - Product manager/owner - Tech lead
Consultancies		
- Teams expected to follow specific a methodology (mostly) - Suggested methodology, but teams can choose (rarely)	1. Scrum 2. No "formal" methodology	A dedicated project manager

Elementi di project management: Project management, Stakeholder, Pert, Gant

Definizione storica di **progetto**: Gestione sistematica di un'impresa complessa, unica e di durata determinata, rivolta al raggiungimento di un obiettivo chiaro e predefinito mediante un processo continuo di pianificazione e controllo di risorse differenziate e con vincoli interdipendenti di costi-tempi-qualità.
Definizione pratica di **progetto**: un progetto è uno sforzo temporaneo intrapreso allo scopo di creare un prodotto, un servizio o un risultato unici.

I **deliverable** sono i risultati del progetto, in alcuni contesti possono anche essere chiamati **outcome**. Possono essere: un prodotto o manufatto che viene prodotto, quantificabile, che costituisce un prodotto finale o un componente di un prodotto. La capacità di erogare un servizio, ad esempio una funzione aziendale a sostegno della produzione o della distribuzione oppure un risultato, come degli esiti o dei documenti.

Per *elaborazione progressiva* si intende lo sviluppo in passaggi successivi e la prosecuzione incrementale del progetto, l'elaborazione progressiva è una caratteristica dei progetti che accompagna i concetti di unicità e temporaneità.
Le strutture organizzative eseguono lavoro per raggiungere una serie di obiettivi, in genere è possibile classificare il lavoro come progetto o come funzioni operative (o processi), anche se le due categorie presentano talvolta aree comuni.
Progetti e processi condividono molte caratteristiche: sono seguiti da persone, sono vincolati da risorse limitate e sono soggetti a pianificazione, esecuzione e controllo. Si distinguono principalmente perché i processi vengono eseguiti in modo continuativo ed hanno natura ripetitiva mentre i progetti hanno natura temporanea ed unica.

Il **project management** è l'applicazione di conoscenze, skill, strumenti e tecniche alle attività di progetto al fine di soddisfarne i requisiti SMART (Specifico/Semplice, Misurabile, Accettabile, Rilevante, Tempificato/Tracciabile). Il project management include: identificare i requisiti, fissare obiettivi chiari e raggiungibili, individuare il giusto equilibrio tra le esigenze di qualità, *scope(ambito)*, tempo e costi, che sono in competizione tra loro. Il project management deve anche adattare specifiche di prodotto, piani e approccio alle diverse aree di interesse e alle diverse aspettative dei vari stakeholder. Il project management ha come obiettivi: dare una visione realistica del progetto durante tutto il suo ciclo di vita, responsabilizzare tutti gli attori coinvolti su obiettivi specifici, evidenziare situazioni critiche e proporre valide alternative in modo tempestivo o comunque in tempo utile, tracciare un quadro previsionale dell'evoluzione futura del progetto, assicurare una coerenza tra gli obiettivi parziali assegnati e quelli generali di progetto.

Il **project manager** è la persona incaricata del raggiungimento degli obiettivi di progetto, lo **stakeholder** sono tutte le persone che hanno interesse in un'organizzazione, in un progetto o in un servizio ecc. Possono essere interessati alle attività, agli obiettivi, alle risorse o ai prodotti. È un termine usato sia per i processi che per i progetti.

Il termine "Project Management" può anche essere utilizzato per descrivere un approccio della struttura organizzativa alla gestione delle funzioni operative. Questo approccio è definito come "gestione per progetti", affronta numerosi aspetti delle funzioni operative (e dei processi che esse formano) sotto forma di progetti per garantire l'applicazione di consolidate tecniche di project management. I sott'insieme del project management sono: gestione complessiva integrata, gestione dell'ambito (scope), gestione dei tempi, gestione della qualità, gestione delle risorse umane, gestione della comunicazione, gestione dei rischi, gestione dell'approvvigionamento.

Il **PERT** (**Program Evaluation and Review Technique)** è il grafo del progetto, un modello logico/funzionale. È composto da nodi per indicare gli eventi (milestone/deliverable) e frecce per le attività oppure si usano riquadri per le attività e frecce per la relazione delle dipendenze.
Un progetto consiste in una serie di attività interdipendenti che devono essere eseguite con una precisa sequenza, con le tecniche PERT/CPM (Critical Path Method) si rappresenta il flusso logico delle attività mediante un reticolo. Con il PERT si hanno durate aleatorie mentre con il CPM durate deterministiche.
Il tipo di reticolo più adottato è quello cosiddetto "ad arco" (tipo AOA – Activity on Arrow) ed è formato da frecce che rappresentano le attività e nodi che rappresentano eventi nel tempo (es. rilascio di deliverable), altrimenti esiste l'AON in cui le frecce indicano i legami tra le attività (riquadri).
Le fasi associate al PERT sono: la pianificazione e costruzione del modello di dettaglio, le stime dei tempi e analisi dei percorsi, la programmazione operativa basata sulle stime fatte e infine il controllo delle operazioni sul progetto in corso d'opera.
Dal PERT è possibile ricavare i componenti del progetto, la rappresentazione della rete associata al progetto, il calcolo della tempificazione, le risorse, l'aggiornamento in corso d'opera e l'esposizione dei risultati (GANTT).
Il grafo di progetto è utile per costruire le basi di una gestione integrata del progetto e per facilitare il livello di comunicazione fra gli esecutori del progetto e per fornire al project manager un quadro logico dell'evoluzione del progetto. Il PERT permette di evitare incomprensioni ed attese inutili durante l'evoluzione del progetto e migliorare il livello di responsabilità reciproca tra gli esecutori delle attività, pone le

basi della costruzione di una pianificazione temporale affidabile e dinamica anche tramite l'uso del *metodo cammino critico (CPM)* ed aumenta il livello di simulazione gestionale del progetto.
In base al tempo e alle risorse richieste da ciascuna attività, si può valutare la durata totale del progetto. Attività con predecessori comuni richiedono il completamento di tutti i suoi predecessori, attività senza tempo di riserva sono critiche.
Il **percorso critico** è la sequenza delle attività presenti nel reticolo logico di un progetto che determina la durata del progetto.
Per applicare il **metodo cammino critico (CPM)** occorre procedere per passaggi successivi:

1. Definire la WBS del progetto
2. In base ad ogni workpackage della WBS identificare le singole attività
3. Valutare una stima delle durate di ogni attività
4. Definire le dipendenze tra le attività e rappresentarle in un reticolo logico
5. Esaminare i percorsi tra attività di inizio e di fine per identificare il percorso critico (durata più lunga)

GLOSSARIO DEL PERT:

- D = durata prevista attività
- ES: data di inizio minima – earliest start
- EF: data di fine minima – earliest finish = ES + D - 1
- LS: data di inizio massima – latest start
- LF: data di fine massima – latest finish = LF – D + 1
- SL: margine di scorrimento LF-EF, quanto può essere in ritardo un'attività senza ritardare il progetto

Il calcolo del **percorso critico** prevede due fasi: la **fase in avanti** che rappresenta la pianificazione al più presto, si calcolano i valori di ES ed EF per ogni attività, la *data di fine minima* del progetto è l'EF dell'ultima attività. La **fase all'indietro** rappresenta la pianificazione al più tardi, la *data di fine minima* del progetto diventa l'LF (data di fine massima) dell'ultima attività. Si calcolano i valori di LS ed LF a ritroso per ogni attività, il valore di LS della prima attività è la *data d'inizio massima* in cui si può iniziare il progetto per finirlo in tempo.

Nel **diagramma di GANTT** (o a barre) le attività del progetto vengono rappresentate come barre su un asse temporale orizzontale, sull'asse verticale possono esservi le attività e/o le risorse. Evidenzia tempi, dipendenze e criticità; permette di monitorare giorno per giorno l'andamento dei progetti. Il diagramma di GANTT è utile per pianificare la tempistica delle attività di progetto, per verificare la fattibilità temporale del progetto e permette a tutti gli interpreti del progetto di avere un quadro generale ed integrato delle date di inizio e fine delle varie fasi del progetto. Il diagramma di GANTT ufficializza a livello strategico le date desiderate (o imposte) di inizio, fine ed eventuali milestone intermedie (master schedule), mentre a livello

operativo ufficializza le date di inizio e fine attese di ciascuna attività elementare (Gantt di dettaglio); infine permette di controllare durante l'avanzamento del progetto i ritardi o gli anticipi rispetto alle date pianificate.

Il **diagramma degli incarichi** indica le durate delle attività del progetto, rappresentate come barre su un asse temporale orizzontale, e le risorse umane sono posizionate sull'asse verticale. È l'agenda degli incarichi per le varie risorse umane coinvolte nel progetto. Bisogna fare delle pianificazioni realistiche: non demoralizzare chi lavora al progetto
, pianificazioni complete ma non concise e soprattutto adattabili entro certi limiti. I progetti lunghi devono essere suddivisi in progetti parziali ben definiti e con risultati chiari.

Nella pratica si determinano le attività del progetto attraverso una WBS, si individuano delle dipendenze logico-temporali fra le attività per poterne definire una sequenza precisa, vengono calcolati i tempi delle singole attività e della somma estesa a tutto il progetto ed infine si mappano i tempi su un calendario, tenendo presenti gli intervalli di disponibilità delle risorse umane a cui le attività vengono assegnate come incarichi. Per risparmiare del tempo si possono impegnare più persone (può portare a degli svantaggi), collegare più fasi, acquistare alcune componenti del progetto oppure aggiungere strumenti.
Il progetto viene controllato mediante la verifica del diario del progetto, l'organizzazione delle riunioni, l'analisi delle tendenze e dei milestone e attraverso le relazioni.
Le informazioni in una *relazione* sono: progetto di riferimento, data e nome dell'autore, breve riassunto attività in corso o eseguite, elenco di tutti i problemi, stadi successivi e note.

La **stima** di un progetto è il processo per assegnare valori previsionali per tempi e impegni delle persone, le fonti più comuni per la stima sono: la metodologia in uso, progetti precedenti e la presenza di sviluppatori esperti. Le stime sono inizialmente degli intervalli e diventano più specifici nel corso del progetto.

La tecnica dei **Function Point (FP)** è usata per valutare le dimensioni dei prodotti software e per misurare la produttività dei team di sviluppo. L'idea base consiste nel quantificare le funzionalità fornite dal prodotto finale in termini di dati e processi significativi per gli utenti finali, è legata più a cosa fare rispetto al come.
I requisiti sono organizzati in cinque tipi: OUTPUT, INTERROGAZIONI, INPUT, FILE E INTERFACCE ESTERNE
I vantaggi principali della tecnica FP consistono nell'essere alquanto oggettiva e abbastanza indipendente dalla tecnologia utilizzata nello sviluppo.

Per l'**identificazione dei task** esistono diversi approcci, l'utilizzo di metodologie esistenti o di analogie o in alternativa l'approccio top down, ovvero suddividere attività in task più piccoli e dettagliati. Un'altra alternativa è descrivere i task come Work BreakDown Structure.

È necessario far corrispondere le abilità alle necessità del progetto ogni volta questo sia possibile.
Considerare le abilità tecniche e quelle interpersonali, bisogna inoltre considerare che i livelli di staffing cambieranno durante il progetto ed aggiungere personale overhead non è sempre produttivo. Bisogna lavorare sulla motivazione, utilizzare ricompense economiche con cautela oppure ricompense intrinseche come il riconoscimento, promozione, possibilità di acquisire nuove abilità etc. Inoltre, bisogna considerare anche i de-motivatori come l'assegnazione di scadenze non realistiche, mantenere cattive condizioni di lavoro, prendere una decisione importante senza l'input della squadra etc. È importante garantire una performance del gruppo, assicurandosi che la squadra comprenda il progetto e gli obiettivi, stabilendo procedure operative e assicurarsi che i membri della squadra arrivino a conoscersi.

Quando si parla di progetti bisogna fare attenzione allo scope creep, per minimizzare la pressione dello scope creep è possibile utilizzare JAD e la prototipazione. Può essere utile implementare un processo formale per l'approvazione di modifiche e/o posticipare requisiti addizionali come future estensioni al sistema.

Le tecniche per la stima dei tempi possono rivelare che il progetto richieda più tempo di quanto disponibile, il **timeboxing** consiste nell'assegnare un periodo temporale fisso ad ogni attività pianificata. Aiuta in queste situazioni per impostare una scadenza ravvicinata ma realistica, identificare il nucleo dei requisiti funzionali essenziali, permette alla squadra di mettere a fuoco le funzioni essenziali e aggiungere in seguito le altre funzioni.
Quando una scadenza viene mancata non bisogna assumere che si possa recuperare, l'UNICA situazione in cui si può avere più tempo è quando il resto del progetto è più semplice della parte precedente E il resto del progetto è più semplice di quanto si era stimato inizialmente.
Valutare la complessità del resto del progetto per determinare il corretto aggiustamento delle schedule anche perché aggiungere personale non è sempre il modo giusto per gestire gli scivolamenti delle schedule.

Capitolo 4 – Analisi dei sistemi informativi

L'obiettivo della **fase di analisi** è quello di sviluppare una comprensione chiara dei requisiti del nuovo sistema utilizzando le abilità del pensiero critico per determinare le vere cause dei problemi ed applicare le conoscenze dei SI e del business per delineare i modi per risolvere i problemi nel nuovo sistema.
L'*analisi dei requisiti* (*requirement engineering*) è l'analisi business orientata alla progettazione di soluzioni IT per la risoluzione delle esigenze che emergono, deve raccogliere i requisiti che servono al progetto IT successivo. Un *requisito* è una capacità necessaria per una delle parti interessate per risolvere un problema o raggiungere un obiettivo, ne esistono diversi tipi:

- *Requisiti di business*: cosa necessita il business?
- *Requisiti degli stakeholder*: quali sono i bisogni?
- *Requisiti della soluzione*: possono essere funzionali (cosa deve fare il software?) o non funzionali (quali sono le caratteristiche che il sistema dovrebbe avere?)
- *Requisiti di transizione*: quali sono le condizioni?
- *Requisiti utente*: cosa deve fare l'utente?

Per quanto riguarda i **requisiti funzionali** si suddividono in: requisiti funzionali espliciti, requisiti funzionali impliciti, requisiti NON funzionali impliciti, requisiti NON funzionali espliciti. I **requisiti funzionali** specificano il supporto che il sistema fornirà all'utente per soddisfare il proprio compito di lavoro. Possono essere *process-oriented* (un processo che il sistema deve eseguire) oppure *information-oriented* (le informazioni che il sistema deve mantenere).
I **requisiti non funzionali** sono le proprietà comportamentali che il sistema deve avere, possono riguardare l'ambiente operativo fisico e tecnico, le protezioni necessarie e le problematiche che influenzeranno il sistema finale.
È necessario *documentare* i requisiti, per farlo si crea un rapporto sulla definizione dei requisiti, un documento testuale che elenca i requisiti in forma sintetica, organizzato in raggruppamenti logici. Uno scopo chiave è definire l'ambito (scope) del progetto (ovvero cosa è incluso e cosa è escluso).
I requisiti di qualità per un software sono non ambiguità, correttezza, completezza, consistenza, modificabilità, verificabilità.
Con *elicitazione dei requisiti* si intende l'utilizzare ogni interazione con dirigenti ed utenti per ricavare interesse, supporto ed entusiasmo per il progetto. Può essere fatta mediante l'utilizzo di interviste, questionari, analisi di documento, JAD (joint application development).
Gli **use case** esprimono e chiariscono i requisiti utente, lo scopo è definire l'interazione attesa tra utente e sistema.
La **business analysis** è un insieme di attività, conoscenze e tecniche necessarie per identificare le esigenze di business di un'organizzazione e determinare migliorie e soluzioni agli eventuali problemi esistenti. La **business process management (BPM)**

è la gestione ottimale di un processo, in modo da ridurre i suoi costi ed aumentare il valore da esso prodotto. Il **business process modeling** è l'analisi e modellazione del processo. La *business architecture* definisce la struttura funzionale di una organizzazione in termini di servizi business e delle informazioni business per essi necessarie. La *business capability* è l'abilità che i servizi business hanno di provvedere le funzionalità business necessarie portando risultati business attesi ed il valore di essi associato.
La *business process improvement (BPI)* consiste in interventi di tipo incrementale e cioè volti al continuo e graduale miglioramento dei processi mentre il *business process reengineering (BPR)* attua degli interventi di tipo radicale e cioè volti al completo ridisegno del processo qualora esso si manifesti del tutto inadeguato agli obiettivi da raggiungere.
BPMN: Business process model and notation è l'evoluzione di UML per il business, inserisce tutti gli elementi necessari per l'analisi dettagliata dei processi, è basato sul punto di vista del Business Analyst e dei suoi clienti. CONTINUA SU APPUNTI IPAD
Il **data flow Diagram** è una modellazione di un sistema informativo in termini di attività o processi e di flussi informativi tra essi. Genera lo *schema funzionale*: descrive il trattamento dell'informazione nella base di dati ed è complementare alla descrizione strutturale fornita dallo schema concettuale.
I data flow Diagram traggono origine dalla teoria dei grafi e sono precedenti all'avvento dei computer. Un sistema è visto come una rete di processi funzionali interconnessi da depositi dati, i DFD enfatizzano le operazioni effettuate sulle informazioni e le dipendenze funzionali che vengono a crearsi fra i vari processi.
I *processi* generano, usano o distruggono i dati e trasformano i dati di un flusso di ingresso in un flusso di uscita. Un *flusso* è uno scambio di informazioni tra processi. Gli *agenti esterni* producono e/o consumano dati, i *depositi* di dati memorizzano informazioni in modo passivo. Un DFD è una quadrupla di insiemi finiti di tutti questi elementi appena elencati; un metodo di rappresentazione è quella "a Grafo", un grafo orientato in cui ogni nodo appartiene a uno dei tre insiemi P, D o A ed ogni arco orientato rappresenta un flusso di dati. Le eccezioni ed il trattamento degli errori viene rimandato in *fase di analisi*, alcune regole da rispettare sono: scegliere nomi significativi per i processi, flussi, depositi ed agenti, numerare i processi, evitare DFD complessi ed inoltre non devono esistere flussi di dati fra due agenti esterni, due depositi ed un'entità esterna ed un deposito.
UML skippato in quanto già visto nel corso "Ingegneria del software".

Capitolo 5 – Progettazione dei sistemi informativi

Nella **fase di progettazione** tutto il lavoro "logico" svolto nella fase di analisi viene trasformato in quello fisico, in questa fase si decide come costruire il sistema e si creano i *system requirement* che descrivono tutti i dettagli tecnici per costruire il

sistema. La **system specification** (**specifica di sistema**) è il deliverable finale dalla fase di progettazione che comunica quale sistema il team di sviluppo implementerà durante la fase di implementazione.
I passi principali della fase di progettazione sono:

- Determinare la *strategia di acquisizione*
- Determinare l'architettura tecnica del sistema
- Affrontare le problematiche di sicurezza e di globalizzazione del sistema
- Effettuare le scelte di hardware e software
- Determinare il modo in cui gli utenti interagiranno con il sistema (GUI, input, e output)
- Progettare i programmi per i processi sottostanti
- Progettare il modo in cui saranno memorizzati i dati
- Creare l deliverable finale (la system specification

Gli elementi della system qualification sono: il design dell'architettura e dell'interfaccia, data storage design, specifiche hardware e software, e molti altri.
I *modi di acquisizione del nuovo sistema* sono:

- *Sviluppo custom* (costruito da 0), i vantaggi sono il poter ottenere esattamente ciò che si vuole, consente la flessibilità e la creatività del team, il nuovo sistema è costruito coerentemente con le tecnologie e gli standard esistenti ed inoltre consente di costruire e mantenere skill tecniche e conoscenze delle funzioni in azienda. Per quanto riguarda gli svantaggi si ha la necessità di tempi e sforzi significativi, spesso il procedimento è costoso e può richiedere abilità non disponibili in azienda.
- *Software acquistato* mediante da ASP (application service provider) o SaaS (Software as a Service) PRO*:* prodotti verificati e collaudati*, risparmi sui costi e sui tempi, si utilizzano le competenze dei vendor.* CONTRO: raramente corrisponde perfettamente alle necessità, i processi organizzativi devono adattarsi al software e si va a dipendere dal vendor per manutenzione e miglioramenti futuri. Non si svilupperanno abilità tecniche e funzionali in azienda, necessità specifiche possono non essere soddisfatte.
- *Esternalizzare (Outsourcing) lo sviluppo ad una terza parte*. PRO: si assume una competenza non disponibile in azienda, può far risparmiare denaro e tempo. CONTRO: non permette di costruire competenze interne, dipendenza dal vendor, opzioni future limitate, prestazioni basate su termini contrattuali, rischio di perdita di informazioni riservate in azienda e di controllo sullo sviluppo futuro. I contratti di Outsourcing sono caratterizzati dai tempi e accordi presi con il vendor, dal prezzo stabilito per il nuovo sistema e dal valore aggiunto (ovvero la differenza fra il valore della produzione di beni e servizi e i costi sostenuti da parte delle singole unità produttive). Alcune linee guida per l'Outsourcing sono: mantenere aperte le linee di comunicazione, definire e congelare i requisiti prima della firma del contratto, selezionare con

cura il vendor e gli sviluppatori, assegnare a qualcuno la gestione della relazione ed enfatizzare requisiti flessibili, relazioni di lungo termine e contratti di breve durata. *Regola delle quattro C*: Competenze, Continuità, Conformità, Costi

La **Request for Proposal (RFP)** sollecita proposte dai vendor, sviluppatori o service provider; spiega il sistema da costruire e i criteri per la selezione tra proponenti, le alternative sono la *Request for information* (più breve e meno dettagliata), *Request for Quote* (si utilizza quando si ha bisogno del solo prezzo). I contenuti di una request for proposal sono la descrizione del sistema desiderato, le necessità o circostanze tecniche speciali, i criteri di valutazione, le istruzioni su come rispondere, la tempificazione desiderata e altre informazioni che possono aiutare il rispondente a creare una proposta più completa o accurata.
Una **matrice di alternative** combina diverse analisi di fattibilità in un'unica matrice, include le fattibilità tecniche, economiche ed organizzative; vengono assegnati pesi per indicare l'importanza relative dei criteri e dei punteggi per indicare in che misura l'alternativa soddisfa il criterio.

Concetto di servizio e i servizi IT

Un **servizio** è un modo per fornire valore ai clienti senza che questi si assumano costi e rischi. Molte aziende tendono sempre più a non vendere prodotto quanto piuttosto servizi "prodotto-centrici", la *servitizzazione* è un portfolio di prodotti e servizi integrati, in cui la messa a disposizione di servizi prodotto-centrici fornisce un principale fattore di differenziazione del mercato. La servitizzazione per l'industria manifatturiera o di altro tipo incoraggia le aziende a rifocalizzare le proprie energie sui propri clienti, prioritizzare i propri bisogni ed obiettivi, e anche a preparare le proprie operazioni interne a fornire una robusta offerta di servizi sul campo.
Esistono tre livelli:

- *Servizi base*: product provision
- *Servizi intermedi*: product repair, condition monitoring, customer help desk.
- *Servizi avanzati*: pay per use, fleet management, availability contract and integrated solution

Un *servizio IT* può essere definito come un insieme di funzioni fornite attraverso sistemi IT nel supportare uno o più aree dell'azienda, può essere costituito da software, hardware e mezzi di comunicazione ma il cliente lo percepisce come un'unica entità.
Una *interfaccia* è definita come un punto dia accesso in cui sono realizzati uno o più servizi disponibili all'ambiente esterno. Molti elementi contribuiscono al servizio, il cliente lo percepisce come una sola entità, occorre conoscere e governare i componenti che contribuiscono al servizio. Il **Service Level Agreement (SLA)** è un accordo sui livelli di servizio, stipulato tra organizzazione IT e cliente nel quale viene descritto dettagliatamente il servizio/i che devono essere forniti e le unità di misura

con cui effettuare le verifiche dei livelli di prestazioni. L'**Operational Level Agreement(ola)** è un accordo stipulato con un reparto interno IT al fine di descrivere dettagliatamente la fornitura degli elementi stabiliti in un servizio. L'**Underpinning Contract (UC)** è un contratto stipulato con un fornitore esterno nel quale viene definita la fornitura degli elementi di un servizio, per esempio la riparazione delle postazioni di lavoro di una linea dati di comunicazione. Tale contratto è simile all'implementazione esterna di un OLA.

Procedure per l'acquisizione dei sistemi informativi

Il *ciclo di acquisizione di una fornitura ICT* consiste nella:

- *Formulazione delle strategie di acquisizione* in funzione del contesto che caratterizza la stazione appaltante;
- *Definizione del contratto* basata sulla descrizione delle attività da espletare, dei prodotti realizzati da dette attività, degli indicatori di qualità caratterizzanti attività e prodotti;
- *Governo del contratto* operato sulle fasi di sviluppo, articolato nelle fasi di progettazione e realizzazione, gestione operativa, manutenzione ed include le attività relative alla migrazione e alla dismissione del prodotto o alla cessazione del servizio.
- *Analisi del grado di soddisfazione* degli utenti della fornitura.

Esistono diversi tipi di **contratti**: contratti per la fornitura di apparecchiature ICT, contratto per la *fornitura chiavi in mano* di un sistema ICT completo, in cui il fornitore si fa carico delle consegne e successiva integrazione delle diverse componenti costituenti il sistema presupposto imprescindibile affinché il sistema sia in grado di funzionare. Contratto per la locazione di apparecchiature ICT, di cui il fornitore mantiene la proprietà, contratto per la locazione di un sistema informatico completo, spesso precostruito, di cui il fornitore mantiene la proprietà ed infine contratto per l'acquisto di programmi software che tramite operazioni di configurazione e parametrizzazione svolgono la funzione di generatori di applicazioni.

La **licenza d'uso di programmi software** è un contratto atipico mediante il quale il fornitore trasferisce al cliente il godimento del software, per un tempo determinato o indeterminato, dietro il pagamento di un canone d'uso che può essere periodico oppure una tantum.

Il **contratto per lo sviluppo di software applicativo** prevede che il fornitore realizzi ex novo programmi applicativi rispondenti a specifici requisiti ed esigenze funzionali del cliente oppure che realizzi delle funzionalità a partire da un pacchetto applicativo precostituito (in questo caso si distinguono appalti informatici nel caso di rapporti tra P.A. ed un'impresa, e contratti d'opera).

Il **contratto per la prestazione di servizi ICT**, afferenti ad una vasta gamma di servizi complessivamente riferibili allo sviluppo ed esercizio di un sistema informativo.

Il **contratto di outsourcing di servizi ICT** in cui il cliente affida al fornitore la possibilità di gestire, in tutto o in parte, il proprio sistema informativo e la connessa organizzazione delle attività al fine di assicurare le esigenze relative ai servizi forniti dal sistema informativo.
Per quanto riguarda la *forma delle gare d'appalto* si hanno a disposizione le seguenti modalità:

- *Asta pubblica*, aperta a tutti e viene aggiudicata all'offerta con il prezzo più basso
- *Licitazione privata*, può essere ristretta ad aziende qualificate invitate, aggiudicata all'offerta più vantaggiosa e basata su specifiche di progetto.
- *Appalto Concorso*, può essere ristretta ad aziende qualificate invitate dall'amministrazione, qualità tecnica e condizioni economiche hanno pesi definiti nella lettera di invito.
- *Trattativa privata* in cui le aziende vengono scelte direttamente dall'amministrazione

Per un appalto è possibile o svolgere una gara unica, suddivisa in fasi, risultando complessa e articolata; in alternativa, è possibile attuare uno *spezzamento dell'appalto* che suddivide la gara in capitolati di minore entità con possibilità di procedure più semplici.
La tecnica dei **Function point (FP)** (definita in IBM) è utilizzata per valutare la dimensione dei prodotti software e per misurare la produttività dei team di sviluppo; l'idea alla base di questa tecnica consiste nel quantificare le funzionalità fornite dal prodotto finale in termini di dati e processi significativi per gli utenti finali, è quindi legata di più al "cosa fare" rispetto al "come fare". I vantaggi principali della tecnica FP consistono nell'essere alquanto oggettiva e abbastanza indipendente dalla tecnologia utilizzata nello sviluppo.
Il **CNIPA (Centro Nazionale per l'Informatica nella Pubblica Amministrazione)** promuove l'attuazione delle politiche formulate dal governo da parte della PA, fornendo supporto nell'uso dell'ICT mediante attività di consulenza e proposta su strategie e azioni puntuali rivolte ai decisori politici, alla PA e agli operatori del settore. Emette normative tecniche a livello secondario, quali linee guida e guide tecniche, svolge una valutazione - ex ante, in itinere ed ex post – delle attività ICT della P.A.
Nel 2012 nasce **AgID**, agenzia per l'Italia digitale, è preposta alla realizzazione degli obiettivi dell'agenda digitale italiana, assicurando il coordinamento informatico dell'amministrazione statale regionale e locale. L'agenzia inoltre promuove e diffonde le iniziative di alfabetizzazione digitale, contribuisce alla diffusione dell'utilizzo delle tecnologie dell'informazione e della comunicazione, vigila sulla qualità dei servizi e sulla razionalizzazione della spesa informatica nella P.A.
Il **Capitolato** è il documento fondamentale di specifica tecnico-economica, inquadra in modo generale tutto il progetto, esplicita i vincoli di compatibilità e di

standardizzazione delle varie parti; in progetti pluriennali pone eventuali clausole di aggiornamento tecnologico ed affronta i problemi tecnici di collegamento tra sotto progetti. È necessario dividere chiaramente in capitoli gli aspetti relativi ad hardware di elaborazione, reti di telecomunicazione, software di base e di ambiente e funzioni applicative. Per le funzioni applicative occorre specificare l'eventuale richiesta dei sorgenti, e bisogna distinguere tra funzioni irrinunciabili, funzioni gradite ma realizzabili anche sotto altra forma o in tempi diversi.
Inoltre, occorre dettagliare le richieste qualitative ed economiche sulla manutenzione e dettagliare le condizioni di formazione.
Per quanto riguarda i *criteri e le metriche di valutazione* occorre: scegliere le voci da valutare, scegliere i pesi da assegnare a ciascuna voce e quale funzione di valutazione utilizzare ed infine utilizzare un foglio elettronico per la tabulazione dei risultati. Di seguito alcune funzioni di valutazione:

- Rapporto tra prezzo offerta e valutazione tecnica ($P = \frac{prezzo\_offerta}{valutazione\ tecnica}$), che deve essere minimizzato. Assegna uguali punteggi ad offerte scadenti (ma economiche) ed offerte tecnicamente valide (ma costose), rende difficile individuare l'offerta più vantaggiosa e pone problemi con punteggi tecnici bassi o incrementali rispetto al minimo richiesto
- $P = \alpha \frac{prezzo\ minimo}{prezzo\ offerta} + \sum punteggio\ tecnico, manutenzione, formazione$, il punteggio per i requisiti non economici raramente riesce a raggiungere il massimo previsto mentre il massimo è certamente raggiunto dal prezzo, il prezzo quindi pesa più di quanto non si desideri realmente; occorre quindi scegliere il valore di alpha in modo da bilanciare i due aspetti.
- $P = \alpha \frac{prezzo\ minimo}{prezzo\ offerta} + \beta \frac{punt\_tecnico}{punt\_tecnico\_max} + \gamma \frac{punt\_manutenzione}{punt\_manutenzione\_max}$, la sommatoria dei coefficienti deve essere pari a 100 e in questo caso sia il prezzo che i requisiti non economici raggiungono il valore massimo poiché tutti sono normalizzati al loro interno

Per il **software** esistono numerose definizioni, occorre una definizione per poter definire forme di tutela giuridica in quanto appartiene alla categoria delle creazioni intellettuali. Quando si parla di "protezione" del software si può intendere una *protezione fisica* (es. sistemi di protezioni fisica del programma come chiavi HW o inserimento di codici o password) oppure una *tutela giuridica* come, ad esempio, il ricorso al diritto di autore sotto forma di tutela di opera dell'ingegno di carattere creativo, applicazione di norme di concorrenza sleale (inefficace perché non può valere nell'ambito privato) o la tutela contrattuale attraverso l'inserimento di clausole che regolino o limitino l'uso da parte di altri utenti (inefficace perché è un accordo fra le parti e non consente di rivalersi contro le terze parti).

Il concetto di architettura dei sistemi

L'**architettura** è insieme dei concetti fondamentali delle proprietà del sistema nel suo ambiente, contenuti nei suoi elementi costitutivi, nelle relazioni che tra essi intercorrono e nei principi del design e nell'evoluzione di essi.
L'architettura esprime una *descrizione formalizzata* e completa di un sistema. Un sistema è un insieme di elementi in relazione fra di loro secondo leggi ben precise che concorrono al raggiungimento di un obiettivo comune.
La descrizione dell'architettura di un sistema è un prodotto, l'architettura è intesa nel suo contesto di appartenenza. È necessario sapere dove il sistema si trova, ovvero a quale ambiente appartiene e come interagisce con esso. La **vista** di un'architettura rappresenta il punto di vista di uno *stakeholder*, esistono 4+1 viste:
Logical View, riguarda la funzionalità che il sistema fornisce agli utenti finali (diagrammi UML con class diagram)
Development View, illustra un Sistema della prospettiva di un programmatore e riguarda la gestione del software (component diagram e package diagram UML)
Process View, tratta gli aspetti dinamici del Sistema e spiega i suoi processi e come comunicano. Si focalizza sul comportamento Runtime del sistema (Acitvity diagram UML)
Physical View, descrive il sistema da un punto di vista di un sistemista. Riguarda la topologia dei componenti software sul livello fisico, così come le connessioni tra questi componenti.
Scenarios (o vista dei casi d'uso), la descrizione di un'architettura viene illustrata usando un piccolo insieme di casi d'uso. Gli scenari descrivono sequenze di interazioni tra oggetti e processi. Sono utilizzati per identificare gli elementi architetturali e per illustrare e validare il progetto architetturale.
Gli *obiettivi della progettazione architetturale* sono assegnare i componenti software del sistema informativo ai dispositivi hardware nel modo più vantaggioso, i principali componenti architetturali di qualunque sistema sono il software e l'hardware.
I *sistemi di software* possono essere divisi in quattro funzioni di base:

- *Memorizzazione dei dati*
- *Logica di accesso ai dati* ovvero l'elaborazione richiesta per l'accesso ai dati.
- *Logica applicativa/business* la logica documentata in modelli BPMN, Use case e nei requisiti funzionali.
- *Logica di presentazione*: la visualizzazione delle informazioni agli utenti e l'accettazione dei loro comandi.

I tre *componenti hardware* primari sono:

- *Client Computer:* dispositivi di input/output utilizzati dagli utenti
- *Serve:* computer multiutente che memorizzano software e dati
- *Rete:* permette l'interconnessione dei computer

Il **TOGAF** è nato come un framework generico e una metodologia per lo sviluppo di architetture tecniche, ma si è evoluto in quadro di architettura per l'intera impresa, le viste TOGAF sono: Business Architecture Views (informazioni, processi ed organizzazione, descrivono i flussi di informazioni business tra persone e processi aziendali), Information Systems Architecture views (dati ed applicazioni, punto di vista dei progettisti e amministratori di database), Technology Architecture views (infrastruttura tecnologica, punto di vista di acquirenti, amministratori, dirigenti del sistema) ed infine composite views (punto di vista di systems administrators, operatori e manager)

Capitolo 6 – Architetture e tecnologie dei sistemi informatici

La struttura di un'applicazione software

La struttura del sistema informatico è composta da **programmi software**, ovvero un'applicazione software avente una sua precisa identità, un insieme di programmi può andare a formare un sistema software atto a svolgere le funzioni associate a uno o più processi business.
Un **processo informatico** è un programma software in esecuzione, il processo informatico usa varie risorse (come memoria RAM, canali di comunicazioni via rete, percentuale del tempo della CPU etc.) per svolgere il proprio compito.
Il **DBMS** è un sistema software che standardizza l'accesso dei processi ai dati, offrendo delle interfacce generalizzate che permettono la condivisione dei dati da parte dei processi informatici e l'indipendenza dei dati rispetto ai processi.
Un database (o base di dati) è un insieme di archivi di archivi di dati gestiti contemporaneamente in modo efficiente ed unitario nel DBMS.
Le applicazioni possono essere divise in base ai ruoli in questo modo:

- **Interattive** se un operatore umano interagisce col programma
- **Macchina-macchina** se due o più applicazioni comunicano tra loro
- **Batch** è un'applicazione che inizia, legge dati, li elabora, li salva e poi termina

In base alle architetture si dividono in:

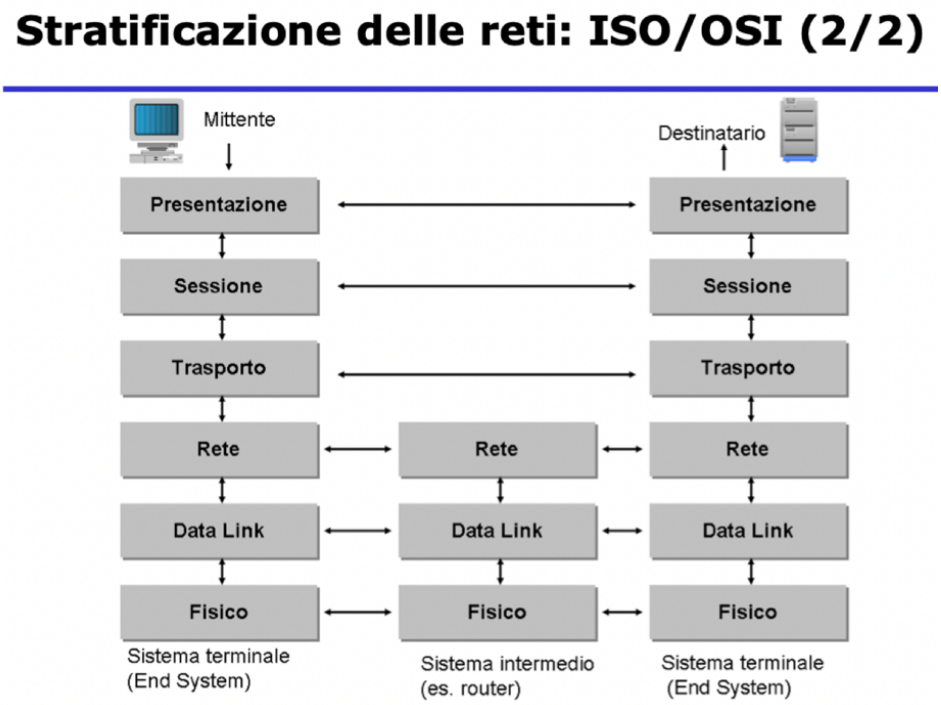
- **Self-consistent**: è un'applicazione completa come MS-Office.
- **Client-server/strutturata:** applicazione suddivisa in moduli, che spesso lavorano su macchine diverse (es. web server e browser).
- **Monolitica:** è un'applicazione Self-consistent non strutturata, che spesso svolge solo un semplice compito.

Un'applicazione è composta principalmente da un'*interfaccia utente (grafica)*, dove vengono presentati dei dati e catturate le interazioni con l'utente; da *regole funzionali (logica business)* che contiene tutte le procedure che compiono le

operazioni in base ai comandi ricevuti dalla UI, e infine i *dati* su cui si deve agire e che devono essere memorizzati.

Le reti entro i sistemi informatici

Il sistema informatico si compone in diverse risorse di calcolo connesse fra loro in rete; quindi, il sistema informatico è basato sulla rete.
Il termine **rete** è nato per indicare un collegamento tra due apparecchiature (sorgente e destinazione) attraverso un mezzo trasmissivo per effettuare una trasmissione di informazioni.
All'inizio le reti erano costituite essenzialmente da terminali remoti collegati ad unità centrali mediante reti locali o linee telefoniche o telegrafiche. L'uso di terminali remoti per l'elaborazione era noto come teleprocessing. La potenza di elaborazione era concentrata in un punto (architettura centralizzata).
La realizzazione di questo tipo di reti era legata a soluzioni *proprietarie*, una soluzione si dice proprietaria quando la realizzazione dipende dal costruttore ed è incompatibile con scelte di costruttori diversi. Molto spesso le specifiche di una soluzione proprietaria non sono pubbliche.
Attualmente per rete di calcolatori si intende un insieme di computer indipendenti, cioè che possono anche lavorare autonomamente ma collegati tra loro in modo da potersi scambiare informazioni. Inoltre, i sistemi sono aperti in modo da poter collegare e utilizzare prodotti di costruttori diversi, ciò rende necessario lo sviluppo di standard comuni. Un fattore importante è la *scalabilità*, cioè la possibilità di aumentare le risorse collegate alla rete in base alla necessità.
La classificazione delle reti odierna è composta da: Local Area Network (LAN), Metropolitan Area Network (MAN). Wide Area Network (WAN).



L’evoluzione tecnologica

L’informatica nelle aziende ha assunto negli anni ruoli molti diversi:

- Negli anni ‘70 il ruolo principale è quello di tecnologia di supporto alla produzione con obiettivi limitati e specifici, le tecnologie utilizzate erano il mainframe + terminali.
- Negli anni ‘80 il ruolo principale è quello di tecnologia di supporto al coordinamento, le tecnologie utilizzate erano i personal computer, reti aziendali proprietarie e risorse di I/O condivise.
- Negli anni ’90 il ruolo principale della tecnologia è quello di supporto alla intermediazione, di uso comune erano internet, intranet, sistemi distribuiti etc.
- Oggi il ruolo principale è quello di tecnologia orientata ai servizi, le tecnologie principale sono: architetture a livelli, architetture cloud, peer-to-peer, web services etc.

L’**outsourcing** è la gestione esterna di elementi non essenziale dell’infrastruttura IT di un’azienda servendosi di Service Provider (SP), i possibili servizi che possono essere gestiti esternamente possono essere: fisici (web hosting, web portal, storage), di manutenzione (apparati di rete, server, PC, …) o globali (gestione globale dei servizi informatici). Bisogna essere in grado di definire in modo molto preciso i compiti e le responsabilità di chi offre il servizio e le relative procedure di verifica delle prestazioni. Oggi sono presenti sul mercato aziende che offrono servizi sia a livello personale che a livello aziendale, queste tipologie di infrastrutture richiedono sistemi con alti livelli di disponibilità e con capacità di adattamento dinamico alle mutevoli condizioni operative. (Esempio: affitto server che offrono servizi di rete o applicazioni scientifiche etc.)
In principio il modello di progettazione dei sistemi informativi era quello a Mainframe (molto diffuso tutt’oggi), composto da un server centrale e terminali “dummy”, fa riferimento a questo modello la topologia di rete a stella.
Con l’avvento del modello client-server si ha un’introduzione di intelligenza nei client: i programmi sul client cooperano con programmi sul server. La topologia tipica di questo modello è la rete a bus, le applicazioni inoltre sono frammentate in DLL, plug-in etc.
Il rischio principale dell’evoluzione di un sistema informativo è nell’avere una *spaghetti integration*: gli “spaghetti” rappresentano i collegamenti dei tool aziendali che si aggrovigliano in infiniti fili di complessità. Uno degli svantaggi principali di questa situazione risiede nella difficoltà nel controllo di processi e flussi che attraversano la catena dei sistemi e Impossibilità di avere una governance centralizzata.

Il mondo del client-server

Un programma server opera su un computer server mentre un programma client opera su una postazione client. L'utente interagisce con il programma client ed il programma client dialoga con il server via rete. Alcuni esempi di architettura client server sono: *2 tier architecture* (la logica dell'applicazione è presente all'interno dell'interfaccia utente sul client o all'interno del database sul server), *3 tier architecture* (la logica o il processo dell'applicazione risiede nel livello intermedio, è separato dai dati e dall'interfaccia utente. I sistemi a tre livelli sono più scalabili, robusti e flessibili.)
Il **middleware** è uno strato software che lega insieme diversi tier.
Le caratteristiche fondamentali del client-server sono: performance e stabilità, complessità e sopportabilità.

Servizi tipici

I **servizi standard di internet** sono: posta elettronica (SMTP), porta 25. Prelievo posta elettronica ricevuta, lettura posta elettronica in remoto, trasferimento file (FTP) porta 20 e 21, World Wide Web (HTTP) porta 80, Terminale remoto, Domain Name System (DNS) porta 53, SecureHTTP (HTTPS) porta 443.
I **protocolli proprietari su internet** sono la condivisione di dischi e stampanti Microsoft/samba, sql*net di Oracle, terminale remoto di Microsoft etc.
L'**informatica distribuita** comprende file server, print server, domain server, groupware server, web server, transaction server, object server etc.
I *file server* sono un servizio di cartelle condivise offerto da un server (solitamente centrale o dipartimentale), lo spazio è fisicamente locato sul disco del server.
I *print server* garantiscono il servizio di stampa condivisa da più postazioni client, è offerto da un server centrale su cui risiedono anche i file temporanei della coda di stampa (non è esente da problemi di safety e security).
I *domain server* si occupano della centralizzazione della gestione degli accessi e dei privilegi mentre l'administration/monitoring servers si occupano della sorveglianza.
I *database server* si occupano della gestione centralizzata dei dati condivisi fra tante postazioni client. (Oracle Server, MS SQL Server, IMB DB2, MySQL etc.)
I *Groupware server* si occupano della gestione della posta elettronica e documenti vari, sono server di sviluppo per il lavoro in gruppo (es. GIT)
I *Transaction Server* sono server che ospitano un sistema di coordinamento di transazioni distribuite (Es. IBM MQ SERIES).
Gli *object server* ospitano sistemi per il funzionamento di oggetti distribuiti (MS DCOM, CORBA.)
I **nodi web** o **server Web** rendono disponibili le informazioni in essi contenute sotto forma di pagine ipertestuali, contenenti documenti multimediali.

Il protocollo HTTP è un protocollo *request/response* il client costruisce ed invia le richieste quando viene specificata una URL mentre il server è in ascolto di richieste HTTP sulla porta 80.

Struttura dell'IT aziendale

Una possibile schematizzazione dell'IT aziendale consiste nel suddividerlo in:
Ambiente esterno, ovvero i prodotti ed i servizi offerti dall'azienda.
Livello Business, le informazioni i processi e l'organizzazione.
Livello applicazioni, i dati e le applicazioni.
Livello Tecnologia, l'infrastruttura tecnologica (sistemi gestione accessi/permessi, sistemi operativi, hardware, infrastrutture di rete sia software che hardware).

Il problema delle compatibilità e le relazioni fra livelli di applicazioni

Una determinata applicazione possiede un look and feel dipendente dalle librerie grafiche usate, opera entro un window manager e se è un'applicazione web utilizza uno standard di HTML e Javascript. I *componenti grafici* spesso dipendono dal linguaggio/ambiente di sviluppo, le librerie relative possono non essere presenti in tutti i sistemi operativi o nelle loro versioni. I *componenti web* possono non essere compatibili con una certa versione di browser/Java/sistema ospite.
Il *protocollo di comunicazione* può richiedere librerie esterne all'applicativo, il *middleware* richiesto può non essere compatibile con il sistema operativo.
Le comunicazioni interprocesso sfruttano meccanismi proprietari, vengono richieste librerie run-time con una versione precisa. Le librerie richieste non sono supportate dal sistema operativo o dalla configurazione presente, ciò può valere anche solo per una parte dell'applicativo.
L'applicativo è strutturato in componenti software e tutti devono essere presenti, vengono sfruttate librerie appartenenti ad altri pacchetti software. L'applicativo sfrutta caratteristiche specifiche di un database, l'applicativo usa stored procedures e richiede le librerie di connessione al database.
Un'immagine di **container** è un pacchetto leggero, isolato ed eseguibile di un software che include tutto ciò che è necessario per eseguirlo: codice, runtime, strumenti di sistema, librerie di sistema, configurazioni.
Il software containerizzato funzionerà sempre allo stesso modo, indipendentemente dall'ambiente (windows, Linux etc.) I contenitori isolano il software dall'ambiente circostante, ad esempio le differenze tra gli ambienti di sviluppo e di staging e aiutano a ridurre i conflitti tra i team che eseguono software diversi sulla stessa infrastruttura.

Docker è un progetto open-source che automatizza lo sviluppo di applicazioni in software containers provvedendo ad un layer addizionale di astrazione e automazione di operating system level virtualization su Linux. La tecnologia Docker è composta da: libvirt: piattaforma di virtualizzazione, LXC: sistemi Linux multipli ed isolati su un singolo host e da file system Layered.
Container multipli possono girare sulla stessa macchina e condividere lo stesso KERNEL con altri container, ognuno su un processo isolato nello spazio utente.
Kubernetes (K8s) è un sistema open source di orchestrazione e gestione di computer, sviluppato da Google, funziona con molti sistemi di containerizzazione (compreso Docker). Le applicazioni moderne vengono sempre più spesso sviluppate con i contenitori ovvero microservizi inseriti in pacchetti insieme alle relative dipendenze e configurazioni. Google ha progettato Kubernetes e poi lo ha rilasciato come open source per aiutare ad alleviare i problemi nei processi di DevOps. L'obiettivo era quello di aiutare con l'automazione, la distribuzione e le metodologie agili per l'integrazione e la distribuzione software. Kubernetes ha reso più facile per gli sviluppatori passare dallo sviluppo alla produzione, rendendo le applicazioni più portatili e sfruttando l'orchestrazione.
DevOps riunisce i team tipicamente in silo diversi: sviluppo e operazioni IT, promette di aiutare i team a lavorare collettivamente e in collaborazione per raggiungere più velocemente i risultati di business. Da un punto di vista tecnologico, DevOps si concentra tipicamente su CI/CD:
Integrazione continua: gli sviluppatori effettuano aggiornamenti costanti nel codice sorgente all'interno di un repository condiviso, che viene poi scansionato e controllato da una build automatizzata, consentendo ai team di rilevare tempestivamente i problemi.
Distribuzione continua: una volta approvato, il codice viene rilasciato in produzione, con il risultato che ogni giorno vengono effettuate molte distribuzioni di produzione
Consegna continua: il software viene costruito e può essere rilasciato in qualsiasi momento, ma con un processo manuale.

Controllare i sistemi: ITIL e TOGAF

Information Technology Infrastructure Library (ITIL) è un insieme di linee guida nella gestione dei servizi IT (IT Service Management) e consiste in una serie di pubblicazioni che forniscono indicazioni sull'erogazione di servizi IT di qualità e sui processi e mezzi necessari a supportarli da parte di un'organizzazione. Il **service catalogue di ITIL** ha due aspetti:

- Il *business service Catalogue (BSC) che* contiene i dettagli di tutti i servizi IT erogati ai clienti, assieme alle relazioni con le unità e i processi di business che fanno affidamento sui servizi IT. È la vista cliente del Service Catalogue. Descrive il servizio a tutti gli utenti/clienti (vista strategica, il "cosa").
- *Il technical Service Catalogue (TSC)* contiene i dettagli di tutti i servizi IT erogati ai clienti, assieme alle relazioni con i servizi di supporto, i servizi condivisi, i componenti e i Configuration Item necessari alla fornitura del

servizio al business. Dovrebbe sostenere il BSC ma non essere parte della vista cliente. (Vista implementativa, il “come”).

I componenti IT ed i servizi con essi forniti sono noti come **configuration item (CI)**, tra i configuration item su cui si basa un servizio IT sono presenti diversi elementi su cui si basa. I CI possono includere l’hardware del PC, i vari tipi di software, i componenti di rete sia attivi che passivi, i server, i processori, la documentazione, le procedure ed i servizi.

Lo storage moderno e le architetture

I **dischi RAID** (Redundant Array of Inexpensive Disks), è una tecnica di installazione raggruppata di diversi dischi rigidi in un computer (o collegati ad esso) che fa sì che gli stessi nel sistema appaiano e siano utilizzabili come se fossero un unico volume di memorizzazione. Le tipologie di RAID principali sono:
RAID 0: suddivisione (*striping)* senza ridondanza, i dati sono divisi equamente tra due o più dischi.
RAID 1: mirroring dei dischi, mantiene una copia esatta di tutti i dati su almeno due dischi.
RAID 2: striping + data check a livello di bit
RAID 3: striping + data check a livello di byte
RAID 4: striping + data check a livello di blocco
RAID 5: striping + parity check distribuito
RAID 6: suddivisione dati tra dischi e parity check distribuito e duplicato
RAID 0+1: sistema a due livelli, in cui un primo livello di RAID 0 viene duplicato ed organizzato in un RAID 1
RAID 1+0 (detto anche RAID 10): sistema a due livelli, in cui vari sistemi RAID 1 vengono uniti in un RAID 0
A livello pratico sono normalmente in uso RAID 1,2,5 e 10.
Gli strumenti per lo storage più comuni sono: interni al server, device di storage con bus dedicato (DAS), Network attached Storage (NAS) e Storage Area Network (SAN).

Sistemi operativi e tecnologie per lo sviluppo di applicazioni: integrazione mediante SOA

EAI: Enterprise Application Integration comprende diversi tipi di integrazione.
L’*integrazione orientata ai dati* avviene a livello di database, può essere real-time o no.
Integrazione orientata a funzioni e metodi, consiste nell’integrazione di applicazioni (A2A), è diretta con paradigma request/response, basata su strumenti di middleware o su codice custom.
L’integrazione *di interfacce utente* (refacing) è la standardizzazione delle interfacce utente entro un unico modello, di solito basata sul browser.

L'*integrazione dei processi business* agisce direttamente sul livello business, non facile da applicare quando vi sono prodotti software con business logic rigida, per essere flessibile conduce implicitamente alla SOA.
La **service oriented architecture (SOA)** è un'architettura di applicazioni, più di un insieme particolare di tecnologie, come i web services, ed è definita indipendentemente da essi. Tutte le funzioni sono definite come servizi indipendenti, con interfacce invocabili ben definite che possono essere chiamate in sequenze definite a formare i processi business. I servizi sono accessibili via rete e descritti attraverso linguaggi standard per la definizione di interfacce. La SOA si pone l'obiettivo di gestire la complessità, la mancanza di flessibilità, le problematiche di granularità legate agli approcci esistenti all'integrazione fra ambienti eterogenei e non.
La struttura della SOA è composta da una topologia di applicazioni software, formata da servizi e clienti dei servizi in relazione 1 a 1 tra loro ma debolmente accoppiati.
Le componenti della SOA sono:

- *Service provider* (fornitore di un servizio), responsabile di creare il servizio, pubblicare l'interfaccia del servizio e provvedere l'implementazione effettiva che realizza il servizio.
- *Service requestor* (cliente richiedente un servizio), componente utente del servizio che deve trovare il servizio per conoscenza diretta o interrogando un repository, deve inviare i dati previsti dall'interfaccia del servizio ed ottenere indietro i risultati.
- *Service broker* (intermediario), registra e categorizza i servizi e si occupa di creare e gestire un repository di servizi.

Le operazioni nella SOA sono:

- Pubblicazione di servizi
- Reperimento di servizi
- Interfacciamento ai servizi

I servizi incapsulano tutta la gestione dello stato, vengono chiamati tramite messaggi inviato con **protocolli inaffidabili.** I messaggi sono documenti di business scambiati per eseguire un processo di business, i servizi si aspettano che gli altri si ricordino delle conversazioni in atto. Una conversazione basata su messaggi richiede che la situazione sia salvata e recuperata quando necessario al processo di business come una conversazione tra persone.
Le tecnologie per SOA sono: *web service basati su SOAP* (basati su XML+RPC su canale di trasporto) e *web service RESTful* (uno stile architetturale, non uno standard).
Per garantire il disaccoppiamento e minimizzare il numero di interfacce il componente intermediario deve assumere un ruolo attivo. L'integrazione dei servizi nella SOA tradizionale avviene mediante l'**enterprise service bus,** una soluzione di

integrazione basata su standard aperti, message-based e distribuita; provvede a Routing, invocazione e mediazione tra i servizi per facilitare le interazioni fra risorse IT distribuite in modo affidabile.
I **microservizi** sono una variante delle SOA che struttura applicazioni in collezioni di servizi, a grana fine con protocolli standard sincroni (HTTP/REST) o asincroni (es. code), per avere massima disponibilità dei servizi è consigliato utilizzare comunicazioni asincrone. I servizi possono essere sviluppati singolarmente ed essere rilasciati individualmente e sostituiti; sono più scalabili grazie ai container e al cloud. Le applicazioni monolitiche invece, sono più complesse e più problematiche da sviluppare, mantenere e sostituire. L'architettura basata sui microservizi è alternativa a quella monolitica, è orientata ai servizi ed è composta da elementi debolmente accoppiati e limitati ad un loro contesto. L'unità dell'architettura è il servizio, ogni unità è modulare, fornisce delle API ed esegue certe operazioni in modo tale da poter sostituire una unità problematica nel minor tempo possibile, inoltre, ogni servizio ha un proprio database dedicato. I PRO di un'architettura basata sui microservizi sono:
i servizi sviluppati da team autonomi, sviluppo continuo e rapido dei servizi, facilità di manutenzione, scalabilità in modo indipendente, servizi rilasciati in modo autonomo, adozione delle tecnologie più appropriate per un servizio ed isolamento dei fallimenti (ovvero fallisce una singola unità e non il sistema).
Per quanto riguarda i CONTRO: la difficoltà nell'identificazione del corretto insieme di servizi che formano il sistema, la difficoltà nel testing e del rilascio del sistema distribuito ed infine una difficoltà nella coordinazione per operazioni che richiedono l'uso di più servizi.
Le **tecniche di migrazione** da monolite a microservizi sono un procedimento complesso, in quanto non esistono metodi standard per l'identificazione dei servizi. Non è necessario che un servizio esegua una sola operazione, è necessario che esegua un insieme di operazioni coerenti tra loro. L'obiettivo finale prevede l'identificazione di servizi debolmente accoppiati, dove le operazioni di un servizio devono cercare di non utilizzare operazioni di altri servizi. Le tecniche di scomposizione per l'identificazione dei servizi principale sono:

- *Scomposizione per capacità di business*: sfrutta la possibilità del business di creare valore, ogni capacità business può essere vista come un servizio. La capacità business si focalizza sugli oggetti del business, una volta identificate le capacità si creano servizi che raggruppano quelle relazionate tra loro.
- *Scomposizione per sottodominio*: approccio di sviluppo orientato agli oggetti basato sulla costruzione di un modello del dominio, basato su Domain Driven Design (DDD) che applica due concetti: sottodomini e limiti del contest. Ogni dominio ha un proprio modello del dominio, mentre con limite del contesto si intende l'ambito trattato.

Gli svantaggi della scomposizione consistono nei problemi di latenza durante la comunicazione tra i servizi, la disponibilità ridotta a causa delle operazioni interprocesso sincrone, la consistenza dei dati tra i vari servizi e la difficoltà nell'ottenere una vista consistente dei dati. Esempi di SOA moderna sono Microsoft Azure ed Amazon AWS.

Dai Cluster al Cloud Computing

I **cluster** sono una classe di architetture di elaborazione parallele che si basa su un insieme di elaboratori indipendenti cooperante per mezzo di una rete di interconnessione e coordinato mediante un sistema operativo che lo rende in grado di operare su di un singolo workload.
Il **commodity cluster** è costituito da nodi di elaborazione commerciali in grado di operare in modo indipendente collegati tramite una rete di interconnessione (LAN O SAN) anch'essa di tipo COTS. Un cluster può essere utilizzato in molti modi, ad esempio per elevate prestazioni su di un singolo problema, elevata disponibilità mediante la ridondanza dei nodi. Elevata banda ottenuta dalla molteplicità dei dischi e dei canali di I/O. I cluster offrono un buon rapporto prestazioni/prezzo ed un rafforzamento del ruolo del personale interno: i sistemi cluster ad elevate prestazione utilizzano moduli elementari che derivano dai sistemi di uso personale. L'amministrazione, la manutenzione HW e SW richiedono competenze maggiormente disponibili sul mercato. Si ha un maggior senso di controllo del sistema e della sua evoluzione, inoltre l'integrazione dei cluster non richiede moduli specializzati. Per la prima volta utenti e venditori hanno a disposizione un'architettura parallela stabile e indipendente, questo garantisce persistenza a lungo termine dell'architettura e giustifica investimenti software.
Un altro vantaggio dell'architettura cluster risiede nella *scalabilità*: nuove infrastrutture di rete locale utilizzate consentono di collegare tra loro un numero molto elevato di nodi senza inserire colli di bottiglia hardware significativi e mantenendo la stessa impostazione architetturale. L'architettura cluster permette anche una grande flessibilità della configurazione e degli aggiornamenti, ed un'alta *availability* (disponibilità), ortogonale alle alte prestazioni.
Un esempio di *High Availability*: i clienti raggiungono il servizio a uno specifico indirizzo IP con due schede di rete per nodo è possibile trasferire dinamicamente l'indirizzo del servizio: da una scheda all'altra (fallimento della scheda) o da un server all'altro (fallimento del nodo), l'operatività dei nodi è verificata mediante un dialogo continuo su un canale dedicato.
Con **virtual machine** si intende un ambiente software emulato su una macchina. La VM emula un ambiente in cui sono installati altri programmi; utilizzando una VM non si ha l'esigenza di disporre di svariate macchine fisiche per i test, un limite all'uso delle virtual machine è dato dalle risorse hardware necessarie a garantire un funzionamento efficace.

Per la virtualizzazione di sistema una singola piattaforma hardware viene condivisa da più sistemi operativi, ognuno dei quali è installato su una diversa macchina virtuale.
Il disaccoppiamento è realizzato da un componente chiamato *Virtual Machine Monitor* il cui compito è consentire la condivisione da parte di più macchine virtuali di una singola piattaforma hardware. Ogni macchina virtuale è costituita oltre che dall'applicazione che in essa viene eseguita, anche dal sistema operativo utilizzato. Il VMM garantisce quindi un *isolamento* tra le VM e la *stabilità* del sistema.
Abbiamo due possibilità per i VMM:

- *VMM di sistema*: le funzionalità di virtualizzazione vengono integrate in un sistema operativo leggero, costituendo un unico sistema posto direttamente sopra l'hardware dell'elaboratore, è necessario corredare il VMM di tutti i driver necessari per pilotare le periferiche. Le componenti di una VMM di sistema sono: l'*host*, piattaforma di base sulla quale si realizzano macchine virtuali, comprende la macchina fisica e l'eventuale sistema operativo ed il VMM. Il *guest* è la VM, comprende le applicazioni ed il sistema operativo
- *VMM Ospitato*, viene installato come un'applicazione sopra un sistema operativo esistente, che opera nello spazio utente e accede l'hardware tramite le system call del S.O. su cui viene installato.

Si parla di *consolidamento della struttura* quando si centralizzano vari servizi su un unico server: i differenti servizi vengono eseguiti da varie virtual machine eseguite dalla stessa macchina fisica. Questa soluzione permette un razionale uso dell'hardware, ottimizzando la gestione delle risorse in funzione delle esigenze istantanee. La **virtualizzazione** riduce il tempo per attivare nuovi servizi, ma il tempo da de dicare alla gestione cresce; un possibile problema è il calcolo dei costi delle licenze software.
Il **virtual server** è un computer server operante entro una virtual machine, può essere spostato da un server fisico ad un altro; il ripristino delle sue funzionalità quasi sempre è il semplice ripristino dell'immagine. La **desktop virtualization** è una tecnologia che separa l'ambiente desktop di una posizione di lavoro con le sue applicazioni dal client fisico usato per accederlo, le postazioni fisiche mostrano in desktop remoto i desktop degli ambienti di lavoro. Da un punto di vista di consumo energetico, sebbene il numero di server fisici sia ridotto aumenta comunque l'utilizzo di CPU: il centro di calcolo deve essere riconfigurato per evitare di sprecare aria condizionata su spazi vuoti. E' possibile clonare le VM e trasferirle da una macchina all'altra facilitando le strategie di disaster recovery e riducendo i rischi di perdite di dati e indisponibilità dei servizi. Il **grid computing** è un modello di calcolo parallelo e distribuito che può significare: la creazione di un supercalcolatore virtuale composto da un insieme di sistemi di calcolo collegati in rete che operano di concerto per risolvere problemi di calcolo di grandi dimensioni; calcolo e spazio di

memoria offerto come servizio da un insieme limitato e controllato di risorse ad un insieme limitato e controllato di utenti.
L'**utility computing** è un modello di business per fornire risorse di calcolo, l'utente acquisisce l'uso di una risorsa da un fornitore di servizi e paga per la quantità di risorse utilizzate. Il vantaggio principale è economico, cioè l'utilizzatore paga solamente quello che usa e quando lo usa, si è spesso in presenza di sovradimensionamento dell'hardware necessario in media.
Il **cloud computing** è il nuovo concetto introdotto dal mercato dell'informatica per identificar è il modo con il quale verranno gestiti i servizi informatici in futuro, è un modo di mettere a disposizione e usare risorse (hardware e software) esclusivamente tramite la rete che promette di metterne a disposizione una quantità che cresce in modo indefinito. Il Cloud computing richiede e sfrutta nuove tecnologie che sono a disposizione (nuovi processori, tecnologia di virtualizzazione, nuove architetture di storage distribuito, reti di accesso ad internet a banda larga). Cloud computer è un sistema computazionale parallelo e distribuito composto da una collezione di computer virtuali interconnessi che sono dinamicamente provvisti e presentati come una o più risorse computazionali unite basati su *service-level agreements (SLA)* stabiliti mediante una negoziazione tra provider di servizi e consumatori.
Le tipologie di Cloud Computing principali sono:

- **BaaS/BPaaS: Business (process) as a Service**, ogni processo business (stampa, pagamenti, e-commerce) è offerto come servizio ed è accessibile da una o più interfacce web abilitate. Esempi:
- **SaaS: Software as a Service**, le applicazioni risiedono nel "top of the cloud stack", i servizi proposti da questo layer sono accessibili dall'utente mediante dei portali Web. Il consumatore non controlla l'infrastruttura cloud (network, servers, sistemi operativi, o anche alcune caratteristiche dell'applicazione stessa) esempi: Dropbox, Shopify, Microsoft office 365
- **DaaS: Data as a Service,** consegna di dati specifici come business data spesso integrati con altri servizi cloud. Esempi:
- **PaaS: Platform as a Service**, un livello più alto di astrazione per rendere un cloud facilmente programmabile, offre un environnement su cui gli sviluppatori possono creare e rilasciare applicazioni senza necessariamente sapere quanti processori o quanta memoria l'applicazione stia usando. Esempi: Heroku, Windows Azure
- **IaaS: Infrastructure as a Service**, la capacità offerta al consumatore di utilizzare risorse fondamentali di computazione dove sia possibile sviluppare ed eseguire software arbitrari (che possono includere anche sistemi operativi o applicazioni). Esempi: IBM Cloud, Google Cloud Platform

- **HaaS: Hardware as a Service,** simile al leasing, l'hardware che appartiene ad un managed service provider è installato sul sito del consumatore ed un Service Level Agreement (SLA) definisce le responsabilità di entrambe le parti. Esempi:
- **FaaS: Function as a Service**, una categoria di servizi di cloud computing che fornisce una piattaforma che consente ai clienti di sviluppare, eseguire e gestire le funzionalità delle applicazioni senza la complessità di costruire e mantenere l'infrastruttura tipicamente associata allo sviluppo e all'esecuzione di un'applicazione. Esempi: Oracle Cloud FN, AWS Lambda

I *modelli di deployment* del cloud sono: *private cloud*, l'infrastruttura cloud è offerta per uso di una singola organizzazione comprendente clienti multipli. *Community cloud*, in cui l'infrastruttura cloud è offerta per uso esclusivo da una community specifica di clienti di organizzazioni diverse ma condividono degli aspetti comuni (missions, requisiti di sicurezza, privacy etc.) Il *public cloud* in cui la struttura è offerta per open-use al pubblico generale. L' *hybrid cloud* è una composizione di due o più tipologie di infrastrutture cloud come un'unica entità.

Le caratteristiche principali del Cloud Computing sono: scalabilità nelle performance, scalabilità nei costi, *cloud self provisioning* e *auto provisioning* (configurazione online del sistema da parte dell'utente o automatica ed un adeguamento real-time delle performance). Il **service level agreement (SLA)** strumenti contrattuali con cui si definiscono le metriche di servizio che devono essere rispettate dal provider nei confronti dei propri client/utenti, pena rimborsi accreditati. Le componenti del SLA per servizi cloud sono:

- la *garanzia del servizio*, specifica le metriche che un fornitore si sforza di soddisfare in un periodo di tempo di garanzia del servizio
- Il *tempo* di garanzia del servizio descrive la durata per cui deve essere s soddisfatta la garanzia del servizio
- La *granularità* del servizio di garanzia descrive la scala delle risorse su cui un fornitore specifica una garanzia del servizio
- *Esclusioni del servizio di garanzia* sono le istanze che sono escluse dai calcoli delle metriche del servizio di garanzia.
- *Credito del servizio* è l'ammonto accreditato al cliente se il servizio non è soddisfatto.
- *Misurazione e segnalazione delle violazioni del servizio* descrive come e quali misure e segnala le violazioni del servizio di garanzia.

I grandi provider mondiali di Cloud Computing sono: Amazon AWS, Google, Microsoft Azure, IBM, Oracle mentre in Italia abbiamo Tim, Aruba, FabbricaDigitale, Enter e molti altri.

Edge computing è un sistema di architettura orizzontale, che distribuisce la computazione, lo storage, il controllo e le funzioni di networking più vicino agli utenti attraverso un continuum cloud-to-thing. È importante per applicazioni di IoT.